I/O

Christoph Niethammer

University of Stuttgart
High-Performance Computing-Center Stuttgart (HLRS)
www.hlrs.de

I/O

- Motivation
- I/O subsystem
- I/O methods
- Parallel I/O

Motivation

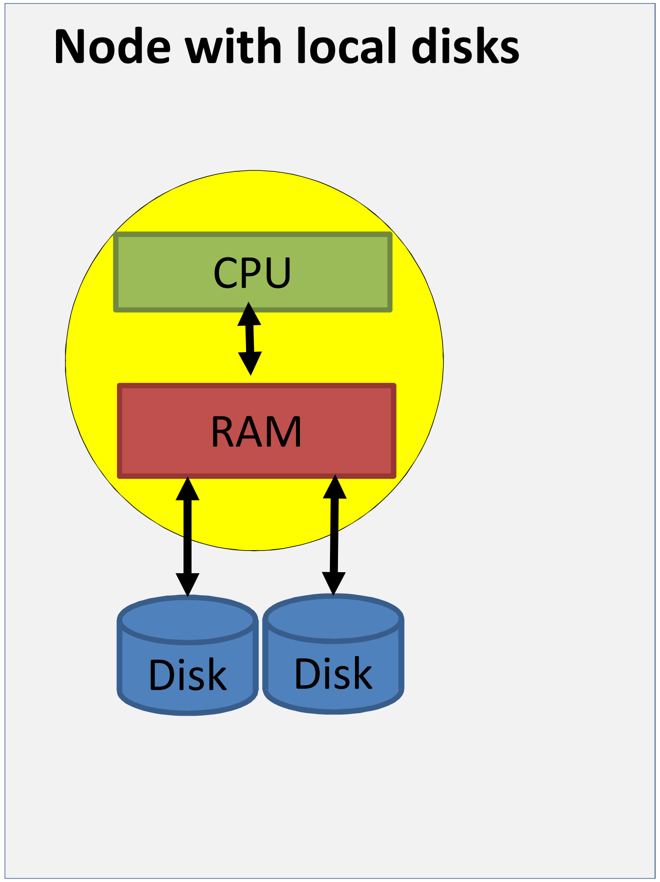
- I/O use-cases:
  - input data
  - checkpointing
  - output data
  - ‘caching‘ of data which do not fit in memory

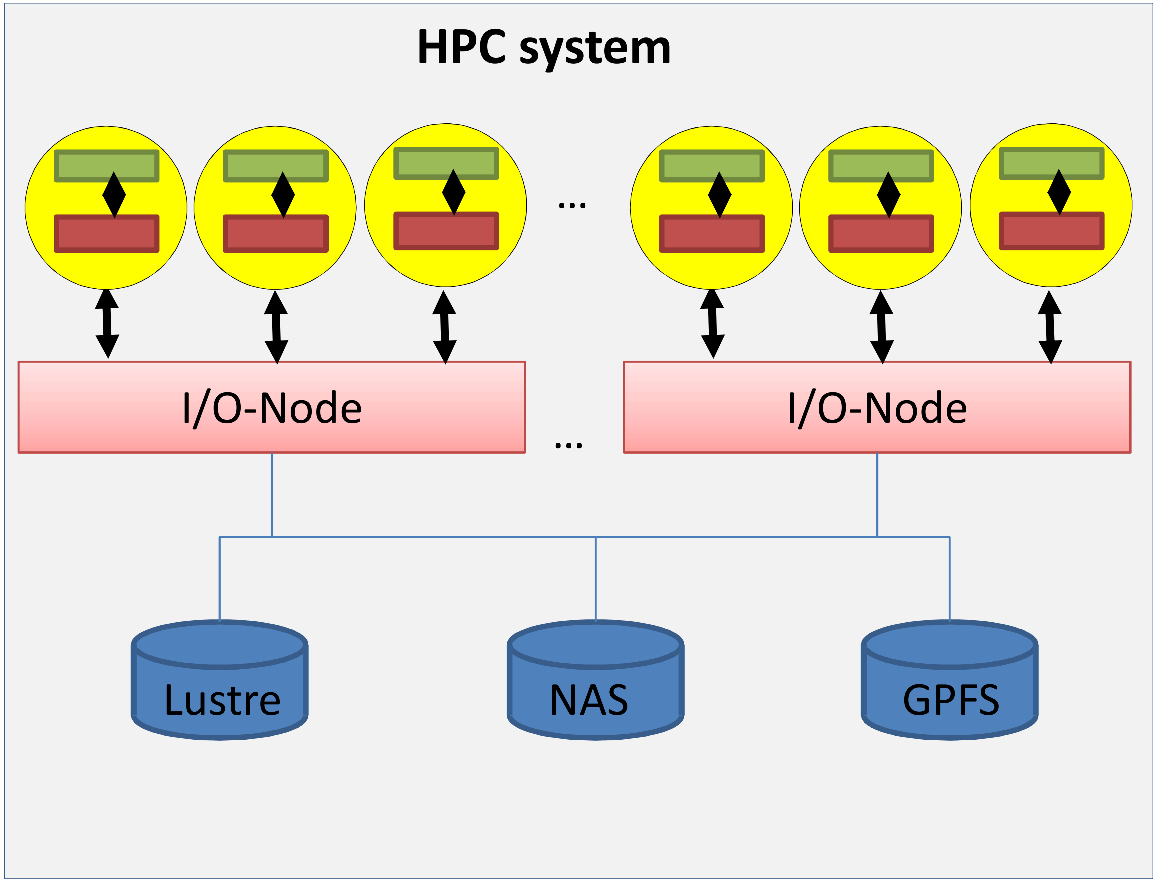
Motivation

- **Moor‘s law** for compute performance is
  “Kryder's law“ for I/O capacity
  BUT **not valid for I/O performance!**

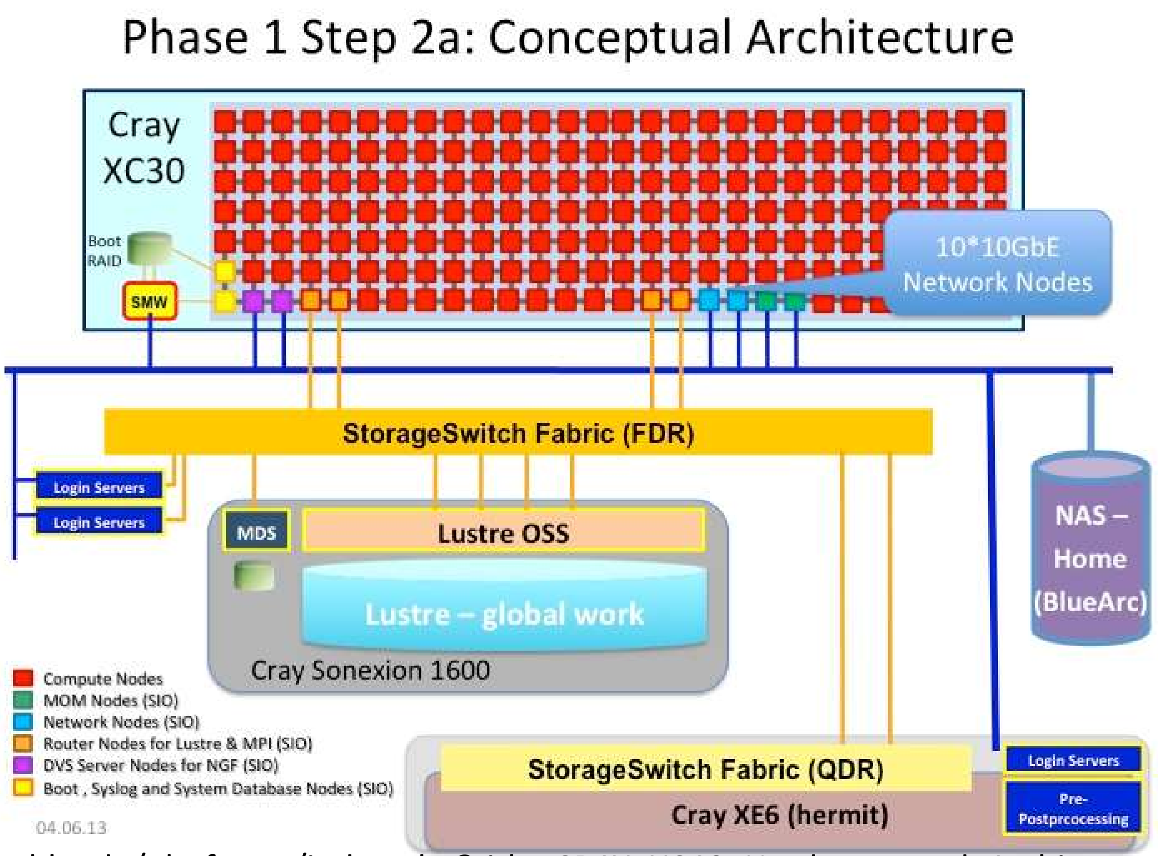
- If I/O takes 0.1% of your serial program →
  Amdhal‘s law → $S_{max}=1/0.001=1000$!

I/O subsystems





HPC system I/O infrastucture (HLRS Hornet)



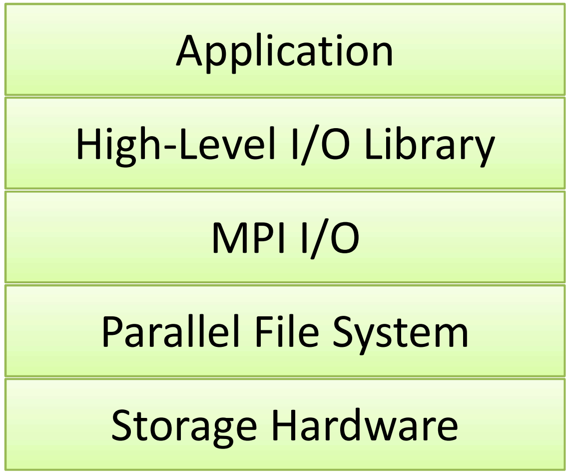
https://wickie.hlrs.de/platforms/index.php?title=CRAY_XC40_Hardware_and_Architecture&oldid=4098

I/O strategies

- Serial I/O via aggregator process:
  →does not scale
- File per process
  →file system limiting
- Parallel I/O on shared file
  →data layout of file very important

I/O layers

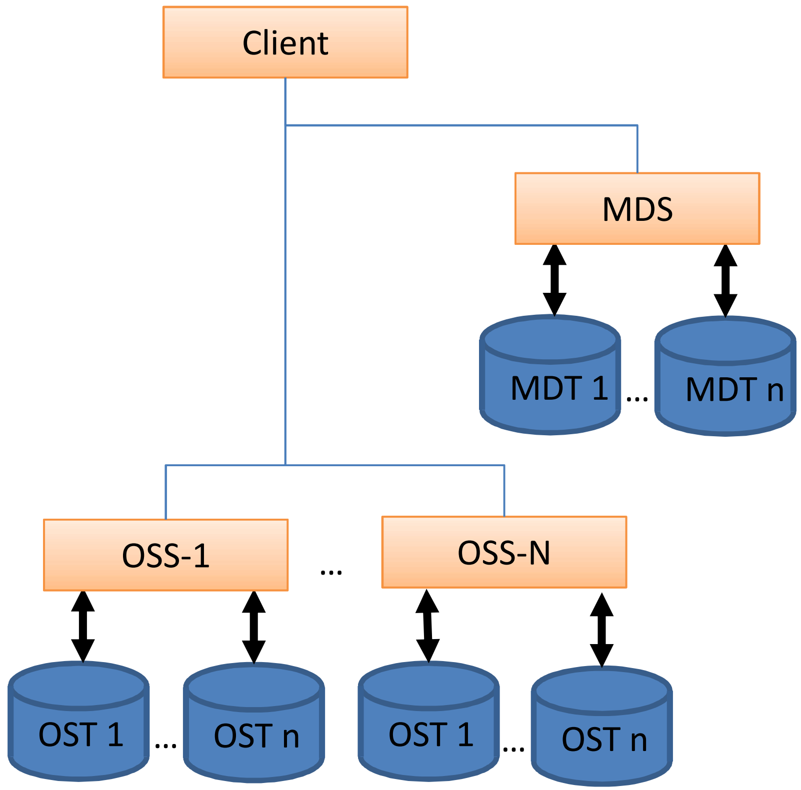
- parallel file system (Lustre, GPFS)
  manages efficient (parallel) data access
- MPI I/O
  manages access by multiple processes
- high-level I/O library (NetCDF, HDF5, ADIOS, …)
  map application data structures to portable file formats



Parallel file system: Lustre

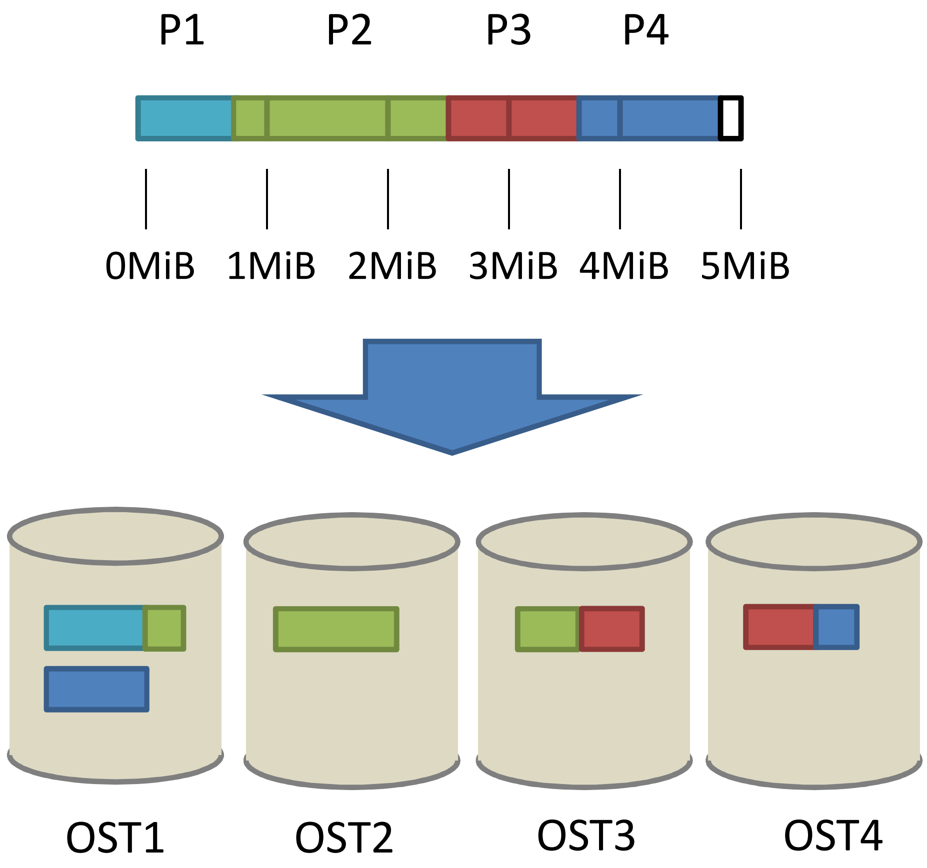
- **Metadata Server** (MDS) stores metadata (filename, file layout, access time, …, directory structure) in one o more **Metadata Targets** (MDT) and opens and closes files
- **Object Storage Server**(OSS) provides file service, and network request handling for one or more local **Object Storage Targets** (OST)
- **Client** handles File access:
  - filename lookup on the MDS
  - interprets the layout in the logical object volume (LOV) layer, and maps offset and size to one or more OSS (→ file stiping)
  - locks files

File I/O done between OSS and nodes



Lustre File striping

- Sequential write of 4 processes



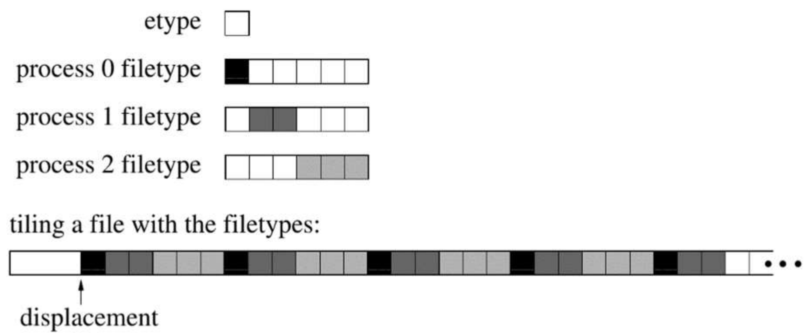
- File is striped to 4 OST with stripe size 1MiB

NOT STRIPE ALIGNED!

MPI I/O

- Independent I/O
  - processes write independent of each other: MPI_File_write(), MPI_File_read()
  - POSIX like with supports for MPI derived datatypes
- Collective I/O
  - All processes have to participate at the same time: MPI_File_write_all(), MPI_File_read_all()
  - Allows the MPI library to perform I/O optimizations

- Uses File Views to describe the data file format



[MPI: A Message-Passing Interface Standard, Version 3.0]

- Uses different binary data layouts: native, internal, external32

NetCDF (Network Common Data Form)

- self-describing data format
- machine-independent
- designed for array-oriented scientific data
- Included in HDF5

```
netcdf simple_xy {
    dimensions:
              x = 6 ;
              y = 12 ;
    variables:
              int data(x, y) ;
    data:
     data =
      0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,
      12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23,
      24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35,
      36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47,
      48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59,
      60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71 ;
    }
```

HDF5 (Hierarchical Data Format)

- hierarchical, filesystem-like data format
- data accessed using POSIX-like syntax /path/to/resource
- User defined metadata for data groups and datasets
- Can store any kind of data, not only array data

```
HDF5 "dset.h5" {
GROUP "/" {
  DATASET "dset" {
    DATATYPE { H5T_STD_I32BE }
    DATASPACE { SIMPLE ( 4, 6 ) / ( 4, 6 ) }
    DATA {
      0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0
    }
  }
}
}
```

ADIOS (Adaptable IO System)

- Provides different optimized I/O transport methods
- Selection of I/O method with very little modification of the applications
- File format metadata are stored in an external Extensible Markup Language (XML)

Conclusion

- I/O can limit overall performance of the application
- HPC I/O system infrastructure complex
- Understand parallel file system to get best performance
- Parallel I/O via MPI I/O or higher-level libraries